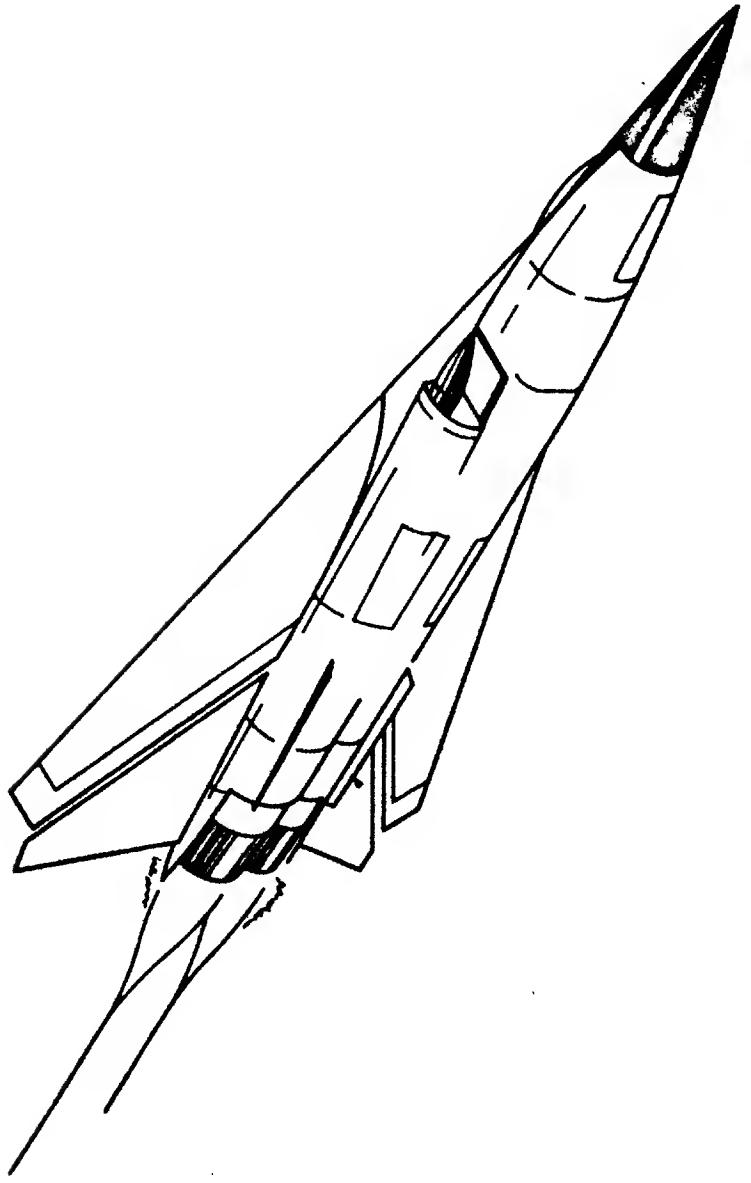
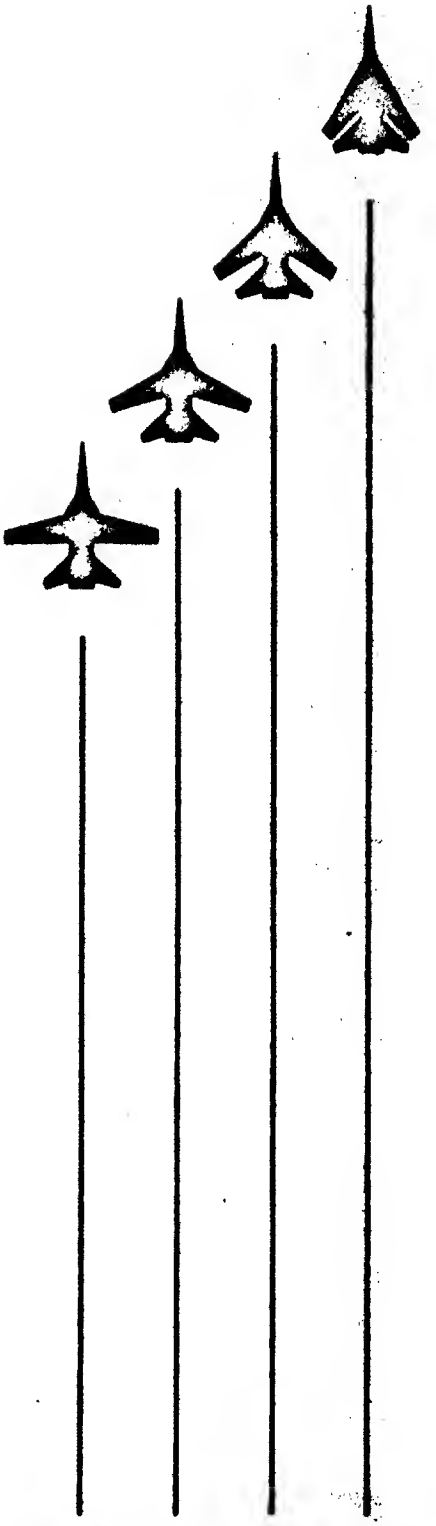
G03308
13 July 1965

AIRBORNE DIGITAL COMPUTER for AN/AWG-9 AIRBORNE MISSILE CONTROL SYSTEM

PROGRAMMER'S MANUAL





CONTROL DATA CORPORATION
GOVERNMENT SYSTEMS DIVISION
3101 EAST 80th STREET, MINNEAPOLIS 20420, MINNESOTA

G03308
13 July 1965

PROGRAMMER'S MANUAL
FOR THE
PHOENIX AIRBORNE DIGITAL COMPUTER

HUGHES AIRCRAFT COMPANY P.O. 44-810564-F-73
STATEMENT OF WORK 65H-9660-29-002, ITEM 7G

PREPARED BY:

D. E. O'Connor

APPROVED BY:

J. I. Malakowsky
Supervisor, Software Design

S. L. McDonough
Program Manager, PHOENIX
Aerospace Computer Project

PROJECT PHOENIX

CONTROL DATA CORPORATION
Government Systems Division

CONTENTS

CONTENTS (Cont.)

LIST OF ILLUSTRATIONS

TABLES

SECTION 1
INTRODUCTION

1.1 PURPOSE OF MANUAL

The purpose of this manual is to provide the information necessary for programming the Control Data Corporation Airborne Digital Computer for the AN/AWG-9 Airborne Missile Control System. Operating instructions are found in the Operation and Maintenance Handbook for the computer. Program loading is accomplished by support equipment supplied with the computer. Program loading information and instructions are found in the operation and maintenance handbook for the support equipment.

1.2 COMPUTER CHARACTERISTICS

The principle characteristics of the computer are as follows. See Figure 1-1.

1) General
   a) Binary, parallel, general-purpose
   b) Integrated circuit logic
   c) Random access NDRO and DRO memories
   d) Scientific-type repertoire
2) Organization
   a) 24-bit instruction length, one instruction per word, single address
   b) Four index registers
   c) 16 external interrupts
   d) "Look Ahead" instruction memory
3) Arithmetic
   a) Fractional, fixed point, two's-complement organization
   b) 12- and 24-bit arithmetic operations
   c) Hardware multiply, divide, search instructions
   d) Execution times

      Add - 2.5 microseconds minimum,
            3.1 microseconds average

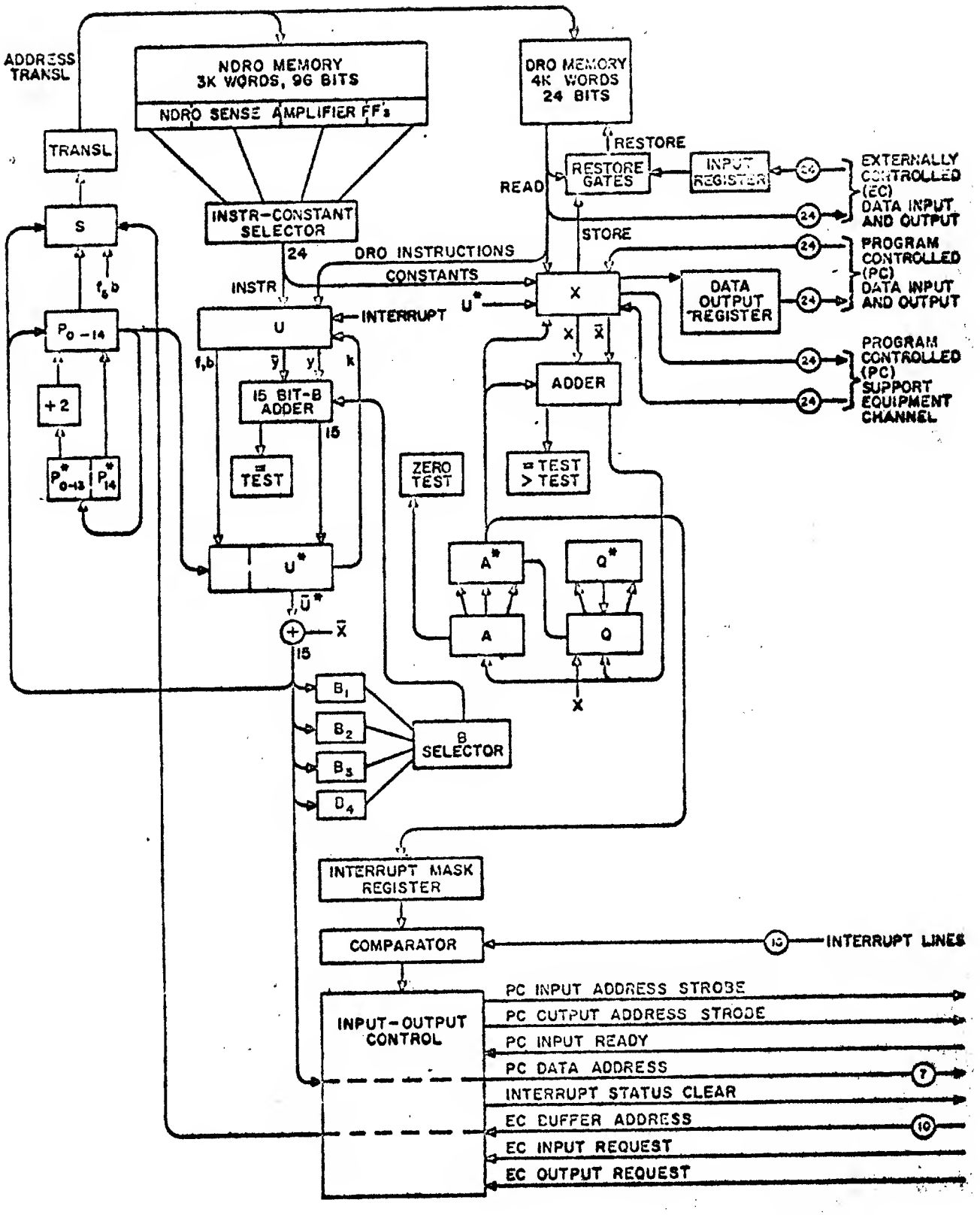


Figure 1-1. Phoenix Central Computer Block Diagram

Multiply - 12 bit, 15.0 microseconds
24 bit, 25.0 microseconds
Divide - 27.5 microseconds

4) NDRO Instruction Memory (thin film)

a) Random-access thin film

b) 12,288 24-bit words

c) 2.5-microsecond cycle time

d) 4-word (96 bit) readout

e) Electrically alterable with support equipment

5) DRO Variable Memory (coincident current magnetic core)

a) Random-access, wide-temperature core

b) 4096 24-bit words

c) 1.0-microsecond access time

d) 2.5-microsecond cycle time

6) Input-Output

a) One externally-controlled input channel (fully buffered)

b) One externally-controlled output channel (fully buffered)

c) One program-controlled input channel

d) One program-controlled output channel

e) One special input and output channel for support equipment

f) Sensing of 16 external interrupts

7) Physical Characteristics

a) Weight: Less than 40 pounds design goal

b) Volume: 1.1 cubic feet

c) Power: 150 watts

d) Cooling: Cold plate

1.3 TERMS AND SYMBOLS

An explanation of terms and symbols used in this manual follows:

Item	Meaning
$\longrightarrow$	Is transferred to
(X)	The contents of X if X denotes a register; the contents of a memory location if X denotes an address
B^b	Index register b if b = 1,2,3, or 4
X_L	Bits 0 through 11 of a register or memory location denoted by X
X_R	Bits 12 through 23 of a register or memory location denoted by X
$(X)\cdot(Y)$	Bit-by-bit (logical) product of the contents of X and Y
$0 \longrightarrow A_N$, if $(M_N) = 1$	0 to each individual bit position of A if same numerical bit position of M = 1
(M_{9-23})	Contents of bit positions 9 through 23 of location denoted by address M
$(A)_i$	Initial contents of A
57.X	X = s or t of word
DRO	Destructive Readout Memory
NDRO	Non-destructive Readout Memory
Op Code	Operating Code (f)

SECTION 2
COMPUTER HARDWARE DESCRIPTION

2.1 COMPUTER HARDWARE

A description of computer hardware follows. See Figure 2-1.

2.1.1 CENTRAL PROCESSOR (CP)

The central processor performs the binary operations associated with the instructions. It also directs the operations required to execute instructions and establishes the timing relationships needed to perform these operations in proper sequence.

2.1.2 NONDESTRUCTIVE READOUT (NDRO)

The NDRO memory is thin film and has random access nondestructive readout. The memory is composed of 3072 words of 96 bits each. This memory capacity is identical to 12,288 words of 24 bits each.

2.1.3 DESTRUCTIVE READOUT (DRO)

The DRO memory utilizes wide temperature cores in a standard stack design. It consists of 4096 words of 24 bits each, alterable under program control. Thus, it can be used for temporary storage, subroutine linkage, counting, etc.

2.1.4 INPUT/OUTPUT SECTION

Digital data is transmitted in parallel to and from the computer with three independent data channels. The programmed data channels transmit or receive data under direct computer program control. The I/O controlled data channel sends and receives data to and from the computer memory independent of computer instruction execution.

2.2 SUPPORT EQUIPMENT (SE)

Support equipment is necessary for program loading but is not necessary for program execution.

The main console contains a control panel, a display panel, and a perforated tape reader.

An auxiliary console contains a typewriter for operator communication with the computer. The typewriter contains a perforated tape punch as well as a slower speed tape reader for off-line operation.

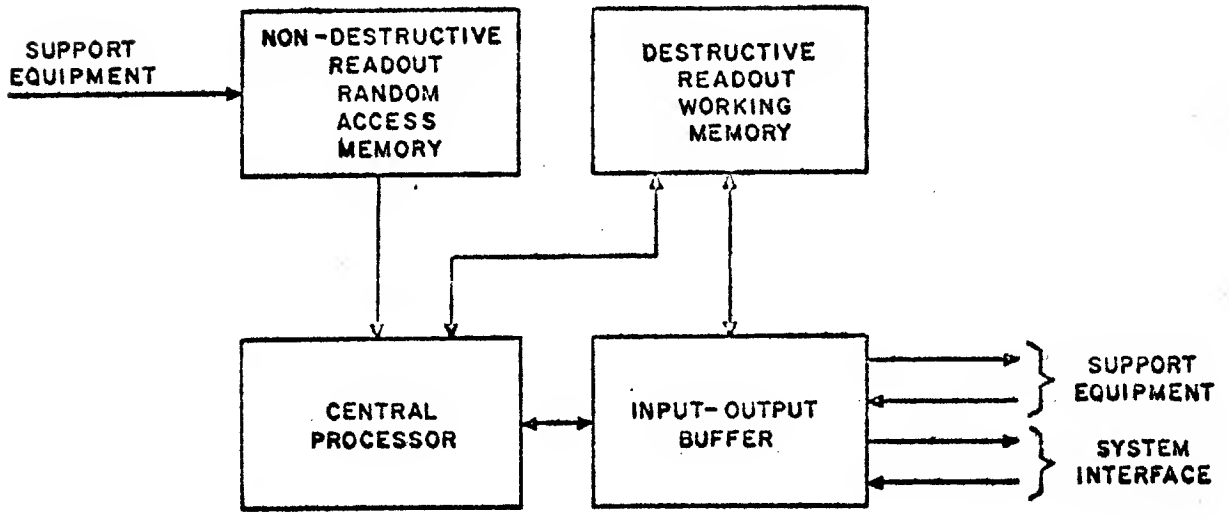


Figure 2-1. Computer Hardware Block Diagram

SECTION 3
FUNCTIONAL DESCRIPTION

Functionally, the computer may be divided into four major sections as listed in Table 3-1.

TABLE 3-1. COMPUTER SECTION

Section	Function
Arithmetic	Performs the arithmetic and logical operations required for executing instructions
Control	Coordinates and sequences all operations for executing an instruction by obtaining the instruction from storage and translating it into commands for the other sections
Storage	Provides internal storage for data and instructions
Input/Output	Provides communication between the computer and the external equipment

3.1 ARITHMETIC SECTION

3.1.1 OPERATIONAL REGISTERS

The operational registers are: 1) A register and 2) Q register. Both registers may load into or be loaded from any of the four index registers (31 to 34) without the use of a storage reference.

3.1.1.1 A Register (Arithmetic)

This 24-bit register is the main arithmetic register. Its contents may be shifted right or left, separately or in conjunction with the Q register to form a 48-bit register.

3.1.1.2 Q Register (Auxiliary Arithmetic)

This 24-bit register is an auxiliary arithmetic register generally used in conjunction with the A register. Its contents may be shifted right or left, separately or in conjunction with the A register to form a 48-bit register.

3.1.2 PRINCIPAL SECONDARY REGISTER

The X register (Exchange) is the principal secondary register. The 24-bit X register is the main exchange register between the A and Q registers and memory as well as between the program controlled data channel and the A register. The X register also contains the multiplicand during multiplication and the divisor during division.

3.2 CONTROL SECTION

The control section directs the operations required to execute instructions. It also establishes the timing relationships needed to perform the operations in the proper sequence. The control section acquires an instruction from storage, translates it, and sends the necessary commands to other sections.

3.2.1 OPERATIONAL REGISTERS

3.2.1.1 P Register (Program Address)

The 15-bit P register is the program address counter. The most significant bit of this register is not used. The remaining 14 bits hold the address of the current instruction. The quantity in P is advanced after each instruction is completed. The amount by which P is advanced is determined by the type of exit the instruction takes. The exits are:

1) Normal exit - The P register is advanced by one.
2) Skip exit - The P register is advanced by two.
3) Jump exit - The P register is set to a quantity defined by the execution address of the jump instruction.

3.2.1.2 U Register (Program Control)

The 24-bit U register is the main program control register and holds the program step while it is being executed. All operations necessary to execute an instruction are governed by the upper 9 bits of this register. The lower 15 bits will contain different types of information depending on the instruction being executed; also associated with the lower 5 bits is a counter which is used in reducing this portion of the U one count at a time, to zero, for certain instructions. Specific functions of the U register are discussed in the instruction repertoire section.

3.2.1.3 Registers B^1 through B^4 (Index Registers)

The 15-bit index registers hold quantities used as address modifiers and hold control quantities for certain instructions as explained in Section 4, Programming.

Computing speed is gained by utilizing the overlap between instruction executions. That is, the index operation (operand address modification) for the next instruction is performed during the execution of the current instruction. The index operation is performed by adding the contents of the selected B register to the 15-bit address portion of the instruction in U. This is possible because the index registers have their own adder.

3.3 STORAGE SECTION

The PHOENIX Airborne Computer memory is composed of two modules: a thin-film nondestructive readout random access module (NDRO) and a magnetic core destructive readout module (DRO). The NDRO memory has a storage capacity of 12,288 words of 24 bits each, while the DRO has a capacity of 4096 words of 24 bits each. These units operate together during the execution of a stored program and can be considered as one 16,384 word storage system.

The location of each word in storage is identified by an assigned number or address. The two modules have the respective addresses:

1) DRO - 00000_8 through 07777_8

2) NDRO -10000_8 through 37777_8

The required memory capacity for the NDRO of 12,288 words with 24 bits per word is achieved by using 3072 words of 96 bits each. Computing speed is thereby increased since only one memory access is required to obtain four instructions. These four instructions are retained by NDRO sense amplifiers between storage references. The cycle time or time for a complete storage reference in the NDRO memory is 2.5 microseconds. Since four words are acquired with one reference, the average time to acquire each word is 625 nanoseconds. The cycle time for the DRO memory is 2.5 microseconds.

3.4 INPUT/OUTPUT SECTION

Information flow to and from the PHOENIX Airborne Computer is handled by the input/output (I/O) section of the computer. The program control channel (PCC), support equipment channel (SEC), and the I/O controlled channel are three independent data channels that transmit or receive digital data in parallel.

The programmed data channels (PCC and SEC) transmit or receive data under direct computer program control. The PCC provides a 24-bit bidirectional data path between the computer and the HAC interface. The SEC provides a bidirectional data path between the computer and the support equipment. The support equipment contains a paper tape reader; a typewriter, register displays, and switches.

The I/O controlled channel sends and receives data under control of the HAC interface equipment. This bidirectional 24-bit channel can transfer data to or from the lower 1024 locations of DRO memory independent of computer instruction execution. See Table 3-2.

TABLE 3-2. SPECIAL ALLOCATIONS

Locations	Use
	DRO Special Allocation
00000-01777	I/O Interface Storage Area
02000-02007	Program Address Register Storage, Subroutine Jump
02010-02017	Program Address Register Storage, Interrupt
	NDRO Special Allocation
10000	Initial Power-up Reset 1
10000	Manual Computer Reset
10001	Power Return Reset 1
37774	Real-Time Clock Interrupt 2
37775	Self-Test Interrupt 3
37776	External Interrupt 3
37777	Power Transient Interrupt 4

3.4.1 INTERRUPT FUNCTION

The interrupt control system of the PHOENIX Airborne Computer provides for testing whether or not certain external conditions exist independently of the main program. There are two major types of interrupts in the computer: 1) special (power transient, real-time clock, self-test) and 2) normal (external).

3.4.1.1 Description of Interrupts

Each of the four interrupts is provided to the computer by the HAC interface. Table 3-3 contains a list of the interrupts.

TABLE 3-3. INTERRUPTS

Interrupt	When Generated
Power Transient	Generated if the prime power to the power supply has gone out of tolerance
Real-Time Clock	Generated if the real-time clock in the HAC interface overflows
Self-Test	Generated if the self-test button is depressed
External	Generated on any one of 16 conditions from the HAC interface

3.4.1.2 Interrupt Processing

Processing of both types of interrupts is accomplished in the following manner. Each interrupt causes the computer to store the contents of the program address register, P, in a location unique to that interrupt. Next, a jump is made to a fixed address unique to that interrupt. See Table 3-2. This address normally contains a jump to the interrupt routine. The interrupt routine takes the necessary action for the condition and then jumps back to the last unexecuted step in the main program. This return to the main program may be accomplished by using the JPI instruction described in Section 4.

3.4.1.3 Computer Interruption

Two times when the computer may be interrupted are: 1) immediately prior to acquisition of a new instruction(s) from storage or 2) following the search of an item during the execution of any search instruction.

The following are times when a new instruction(s) is acquired from storage. Thus an interrupt can be recognized at these times:

1) Prior to acquiring a sequential group of four instructions from NDRO memory.

2) Prior to acquiring an instruction from DRO memory

3) After executing an instruction acquired from NDRO memory that references an operand in NDRO memory

4) After execution of any jump, multiply, or divide instruction or after execution of an instruction taking a skip exit.

3.4.1.4 Interrupt Priority

The special interrupts (power transient, real-time clock, and self-test) have priority over the normal (external) interrupt. A priority also exists within the special interrupts. The priorities are:

1) power transient - top priority

2) real-time clock, self-test - equal priority

The processing of the power transient interrupt cannot be interrupted. The processing of all other interrupts can be interrupted by an interrupt of equal or higher priority.

3.4.1.5 External Interrupt Register

This 16-bit register is located in the HAC interface and is used by the interface to interrupt the computer on any one of 16 conditions.

3.4.1.6 Interrupt Mask Register

This 17-bit register is located in the PHOENIX Airborne Computer. The 2^0 bit is the interrupt inhibit bit; the remaining 16 bits correspond to the 16 bits in the external interrupt register. The programmer can, with the EXF instruction, choose to honor or ignore an interrupt indication from the external interrupt register by selectively setting or clearing the corresponding bits in the interrupt mask register.

3.4.1.7 Interrupt Inhibit Bit

The interrupt inhibit bit can be set or cleared under hardware or software control. When an interrupt is recognized, the interrupt inhibit bit is set by hardware. When set, this bit prevents the recognition of subsequent external interrupts. Thus the routine processing of external interrupts cannot be interrupted by another external interrupt.

3.4.1.8 Recognition of External Interrupt

The recognition of the External Interrupt is accomplished automatically by hardware examination at the times given above on the basis of the logical product of the contents of the external interrupt register and the contents of the interrupt mask register. If a 1 bit is found anywhere in the product, condition for an interruption exists. However, if the interrupt inhibit bit is set, the condition is ignored until the inhibit is cleared. Upon recognition, the interrupt processing as described above begins.

3.4.1.9 Recognition of Special Interrupts

The power transient, real-time clock, and self-test interrupts are examined and recognized automatically by the hardware in the priority and at the times given above. These interrupts are special because they are not disabled by the interrupt inhibit bit but, when encountered, they cause the inhibit bit to be set. Thus these interrupts will take priority over the normal (external) interrupt.

3.4.1.10 Sensing Interrupts

The programmer may selectively sense external interrupts independently of the interrupt mask register by executing EXF instruction that transfers the contents of the external interrupt register to the A register. Executing this instruction automatically clears the external interrupt register.

3.4.1.11 Clearing Interrupts

The special interrupts are self-clearing. The external interrupts are cleared by executing the EXF instruction that transfers the contents of the external interrupt register to the A register. (See paragraph under heading 3.4.1.10.)

3.4.1.12 Interrupt Routine

An interrupt routine must be programmed. The interrupt routine typically:

1) Stores the contents of the operational registers if these registers are to be used in the interrupt routine.

2) Processes the interrupt condition(s).

3) Clears the condition causing the interrupt.

4) Reloads the operational registers upon completing the routine.

5) Returns control to main program.

3.4.2 RESET CONTROLS

The three resets provide a way to force the computer program to restart from fixed locations by setting the program address register, P, to reference these locations. See Table 3-2. The three resets are:

1) Manual computer reset - This reset occurs when the equipment reset switch is depressed.

2) Power return reset - The HAC interface provides this reset when the DC power is within specified tolerance after the occurrence of a power transient interrupt.

3) Initial power-up line - The HAC interface provides a logic 1 when power is first turned on in the computer subsystem. At the termination of the pulse the computer starts execution.

SECTION 4
REPERTOIRE OF INSTRUCTIONS

These instructions pertain to the single address computer which is sequenced by an internally stored program. This program, together with the data to be processed, is contained in a central, random-access memory. Each instruction is a 24-bit quantity which specifies a certain operation to be performed by the computer.

4.1 STANDARD INSTRUCTION WORDS

Typically standard 24-bit instruction words are divided into three parts: a 6-bit operation code; a 3-bit designator; and a 15-bit base execution address. The formats of the standard instruction words are shown in Figure 4-1. Table 4-1 shows the use of each part of the instruction.

The execution address and designator portions of the instruction are discussed in paragraph 4.1.1. The operation codes are discussed in detail in paragraph 4.1.5. The input/output instructions, which have different uses for the lower-order 15 bits of the instruction, are discussed in paragraph 4.2.

Tables A-1 and A-2 in the appendix contain the instruction repertoire.

4.1.1 EXECUTION ADDRESS

The base execution address may be used as:

1) A shift count, k
2) An operand, y
3) An address of an operand, m, in storage

The base execution address may also be modified or left unmodified depending on the index designator b. If unmodified, the execution address is represented by the lower-case symbol, k, y, or m; if the address is modified the symbols are capitalized. The following examples point out the relationship between the unmodified and modified execution address.

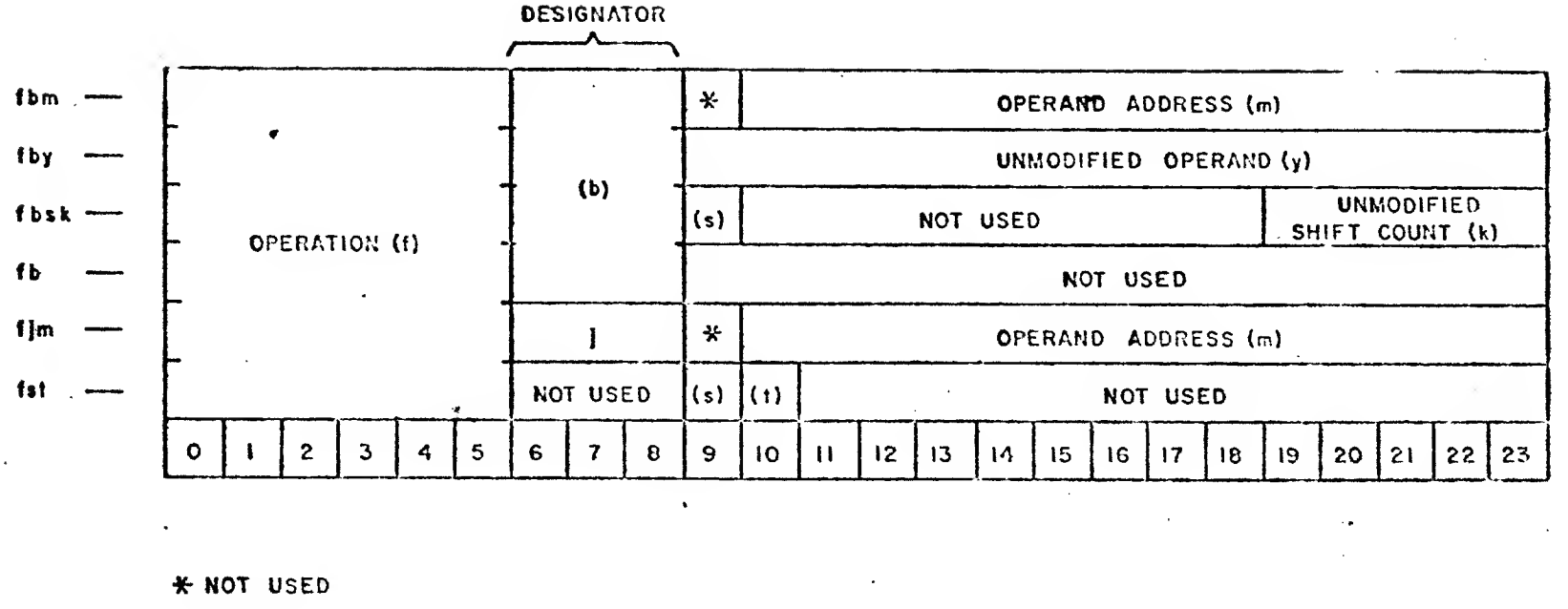


Figure 4-1. Instruction Word Format

TABLE 4-1. INSTRUCTION USE

Instruction Word Part	Numerical Range (octal)	Use
Operation f	00-77	Specifies the operation to be performed.
Designator b		Specifies the index designator whose contents are to be referenced.
	0,5,6, or 7 1-4	No address modification. Relative address modification.
j	0-7	Specifies conditions (refer to jump and stop instructions).
Base		
Execution		
Address k	00000-00037	Unmodified shift count
m	00000-37777	Unmodified operand address
y	00000-77777	Unmodified operand
Sub-operation Code s	0-1	For shift instructions: s = 0, extend sign s = 1, end-off For complement instructions: s = 0, CPA s = 1, CPQ
Complement t	0-1	Specifies 1's or 2's t = 0, one's complement t = 1, two's complement

1) The modified shift count K is represented by

$$K = k + (B^b)$$

where: K = modified shift count

k = unmodified shift count (base execution address)

(B^b) = contents of index register b where $b \neq 0, 5, 6, \text{or } 7$

If the index designator = 0, 5, 6, or 7

then K = k

2) The modified operand Y is represented by

$$Y = y + (B^b)$$

where: Y = modified operand

y = unmodified operand (base execution address)

(B^b) = contents of index register b where $b \neq 0, 5, 6, \text{or } 7$

If the index designator = 0, 5, 6, or 7

then Y = y

3) The modified operand address M is represented by

$$M = m + (B^b)$$

where: M = modified address of operand

m = unmodified address of operand (base execution address)

(B^b) = contents of index register b where $b \neq 0, 5, 6, \text{or } 7$

If the index designator = 0, 5, 6, or 7

then M = m

Note that example 3 is the only case in which the execution address is interpreted as an address of an operand.

4.1.2 ADDRESS MODIFICATION

The two possible modes of address modification are identified by the index designators as follows.

1) b = 0, 5, 6, or 7 — No Address Modification. In this mode the base execution address is interpreted without modification; nothing is added to or subtracted from it (Direct Addressing).

2) b = 1 through 4 — Relative Address Modification. In this mode the base execution address is modified and is equal to the initial execution address plus the contents of the designated index register.

Examples:

1) No Address Modification

f	b	m
LDA	0	address

This instruction is interpreted as load accumulator from the storage location designated by the sum of the base execution address and the contents of the specified index register, B^b. Since b = 0, no index register is designated and m specifies the storage location whose contents are loaded into A.

2) Relative Address Modification

f	b	m	
LDA	1	address	$(B^1) = 00001_8$

In this example, the accumulator is loaded from the storage location designated by the base execution address plus the contents of index register, B^1. Therefore, the contents of the storage location named by the base execution address plus 00001_8 is loaded into the accumulator. $M = m + (B^1)$.

4.1.3 TIMING

Table A-1 in the appendix shows the time required to execute an instruction after a memory reference. Whenever a memory reference is required, the total execution time is 2.5 microseconds

more than that time indicated in Table A-1 with the exception of the instructions referenced to the footnotes in Table A-1. Memory referencing is required:

1) when an instruction is located in DRO,

2). after the last instruction in an NDRO word of four instructions is executed, and

3) when an operand is read from NDRO.

Typically, instructions are stored in NDRO, and since each NDRO 96-bit word contains four 24-bit instruction words only one memory reference is required for each four instructions. If an operand is read from NDRO, the 96-bit NDRO word containing the next instruction must be again accessed. If instructions are accessed from the DRO, a memory reference is required for each instruction since the DRO word length is 24 bits and each DRO word contains only one instruction word.

The instructions referring to the footnote in Table A-1 (multiply, divide, jump, and skip exit instructions) acquire the next instruction from storage as part of their normal operation; therefore, the time to acquire the next instruction is always included in the execution times given in Table A-1. It is never necessary to add 2.5 microseconds for memory referencing when an instruction follows an item referred to the footnote in Table A-1.

4.1.4 SEQUENTIAL EXECUTION OF INSTRUCTIONS

In order to decrease instruction execution time, four instructions are read from storage and then executed sequentially. The addresses of the four instructions differ only in the lower-order two bits; the upper 13 bits of their addresses are identical. The lower-order two bits of the address of the last instruction of the four is 11_2.

Example:

	f	b	m
(10300) =	LDA	0	00310
(10301) =	ADD	1	00210
	(B^1) =		00101_8
(10302) =	. . . . .		

The P register holds address 10300. The storage reference is initiated and the next four instructions are read; the 24-bit word read from address 10300 is entered into U. Computer operation is now dependent upon the interpretation of the 24-bit instruction in U. The execution time for this instruction is 5.0 microseconds, since a storage reference was required.

The operation code, LDA, and the index designator, 0, are translated. The function of the instruction, LDA, is to load the A register with the contents of the designated storage location. Because the index designator is 0, the base execution address is not modified. The translation of the operation code initiated the sequence of the commands which execute the instruction, and the operand in location 00310 is loaded into A.

The contents of P are increased by one, and the next instruction is the 24-bit word read from storage location 10301. This instruction is now translated in U. The ADD instruction causes the quantity in storage location M to be added to the contents of the A register. Since the index designator is not 0, 5, 6, or 7, the contents of the index register are added to the base execution address to form M. Then $M = m + (B^1) = 00210_8 + 00101_8 = 00311_8$.

The contents of storage location 00311 are added to the contents of the A register completing the instruction. The contents of the P register are again increased by one and the instruction at address 10302 is read from storage and executed. The execution time for the ADD instruction is 2.5 microseconds, since a storage reference was not required to get the instruction.

4.1.5 ANALYSIS OF INSTRUCTIONS

The following paragraphs describe each of the standard instructions as they are classified in Table A-1.

4.1.5.1 Shift Instructions

All modes of address modification apply to all of these instructions. If the shift count is greater than 31_{10}, the shift count modulo 32_{10} is used. (K, which is held in U, is reduced by one count for each shift executed. When K=0, shifting stops.) Shifting will be completed before an interrupt request is processed. The maximum possible number of shifts is 31_{10}.

(44) SCA b s k — Scale

Shifts A left end-off until the number in A is normalized or until k = zero. The contents of A are normalized when the left two bits are not alike. The shift count k is decreased by one for each shift of A. Upon termination the count is entered into the designated index register. Since k is a 5-bit quantity the final contents of the designated index register is less than 2^5.

4.1.5.2 Arithmetic Instructions

(00) ADD b m — Add

Adds a 24-bit operand obtained from memory location M to (A). The contents of memory location M remain unchanged.

(04) SUB b m — Subtract

Obtains a 24-bit operand from memory location M and subtracts it from the initial contents of A. The contents of location M remain unchanged.

(20) MUF b m — Multiply Fractional

Forms a 48-bit product of two 24-bit operands. The multiplier must be loaded into A prior to execution of the instruction. The execution address specifies the memory location of the multiplicand. The product is contained in AQ as a 48-bit quantity.

(23) DVF b m — Divide Fractional

Divides a 48-bit fractional dividend by a 24-bit fractional divisor. The 48-bit dividend must be formed in the AQ register prior to executing the instruction. The 24-bit divisor is read from the memory location specified by the execution address. The quotient is formed in Q and the remainder is left in A at the end of the operation. Dividend and remainder have the same sign.

(51) INA b y — Increase A

Adds Y to A. The 15-bit operand Y with its highest-order bit extended in the remaining nine bits is added to A. Y is thus considered a 15-bit signed operand with sign extended.

(42.1) ARE & (42.0) ARS b s k A Right Shift

Shifts contents of A to the right K places. The lower bits are discarded. The largest practical shift count is 23_{10}. Zeros are shifted into the sign bit in the ARE instruction. The sign bit is extended in the ARS instruction.

(41.1) QRE & (41.0) QRS b s k Q Right Shift

Shifts contents of Q to the right K places. The lower bits are discarded. The largest practical shift count is 23_{10}. Zeros are shifted into the sign bit in the QRE instruction. The sign bit is extended in the QRS instruction.

(43.1) LRE & (43.0) LRS b s k Long Right Shift

Shifts contents of AQ to the right K places on one 48-bit register. The A register is considered as having the leftmost 24 bits and the Q register as having the rightmost 24 bits. The lower-order bits of A replace the higher-order bits of Q and the lower-order bits of Q are discarded. The largest possible shift count is 31_{10}. Zeros are shifted into the sign bit of A in the LRE instruction. The sign of A is extended in the LRS instruction.

(47.1) LLE & (47.0) LLC b s k Long Left Shift

Shifts contents of AQ to the left K places as one 48-bit register. The higher-order bits of Q replace the lower-order bits of A. The largest possible shift count is 31_{10}. Zeros are shifted into the least significant bits of Q in the LLE instruction. The higher-order bits of A replace the lower-order bits of Q in the LLC instruction.

(45.1) QLE &(45.0) QLC b s k Q Left Shift

Shifts contents of Q to the left K places. The largest practical shift count is 23_{10}. Zeros are shifted into the least significant bits of Q in the QLE instruction. The higher-order bits of Q replace the lower-order bits in the QLC instruction.

(46.1) ALE &(46.0) ALC b s k A Left Shift

Shifts contents of A to the left K places. The largest practical shift count is 23_{10}. Zeros are shifted into the least significant bits of A in the ALE instruction. The higher-order bits of A replace the lower-order bits in the ALC instruction.

(61) INI b y — Increase Index

Increases (B^b) by the operand y. If the b designator is zero, five, six, or seven this instruction becomes a pass or do-nothing instruction. Relative address modification does not apply to this instruction.

(30) RAD b m — Replace Add

Obtains a 24-bit operand from storage location M and adds it to the initial contents of A. The sum is left in A and is also transmitted to location M.

(03) ADQ b m — Add to Q

Adds a 24-bit operand obtained from memory location M to (Q). The contents of memory location M remain unchanged.

(01) ADL b m — Add Left

Adds a 12-bit operand obtained from the left half of memory location M to the left half of A. The contents of memory location M and the right half of A remain unchanged.

(02) ADR b m — Add Right

Adds a 12-bit operand obtained from the right half of memory location M to the left half of A. The contents of memory location M and the right half of A remain unchanged.

(05) SBL b m — Subtract Left

Subtracts a 12-bit operand obtained from the left half of memory location M from the left half of A. The contents of memory location M and the right half of A remain unchanged.

(06) SBR b m — Subtract Right

Subtracts a 12-bit operand obtained from the right half of memory location M from the left half of A. The contents of memory location M and the right half of A remain unchanged.

(21) MFL b m — Multiply Fractional Left

Forms a 36-bit product of a 12-bit multiplier and a 24-bit multiplicand. The 24-bit multiplicand must be loaded into A prior to

execution of the instruction. The 12-bit multiplier is obtained from the left half of memory location M. The product is contained in A and the left half of Q as a 36-bit quantity.

(22) MFR b m — Multiply Fractional Right

Forms a 36-bit product of a 12-bit multiplier and a 24-bit multiplicand. The 24-bit multiplicand must be loaded into A prior to execution of the instruction. The 12-bit multiplier is obtained from the right half of memory location M. The product is contained in A and the left half of Q as a 36-bit quantity.

(31) RSB b m — Replace Subtract

Subtracts (M) from (A) and places the result in both the A register and location M.

(32) RAO b m — Replace Add One

Replaces the operand in storage location M with its original value plus one. The result is also placed in A.

(33) RSO b m — Replace Subtract One

Replaces the operand in storage location M with its original contents, minus one. The difference is also left in A.

(07) SBQ b m — Subtract from Q

Obtains a 24-bit operand from memory location M and subtracts it from the initial contents of Q. The contents of M remain unchanged.

4.1.5.3 Logical Instructions

All modes of address modification apply to these instructions except CPA and CPQ.

(36) LDL b m — Load Logic Product

Loads A with the logic product of Q and the contents of the designated memory location, M. A logic product is a bit-by-bit multiplication of two binary numbers:

$0 \times 0 = 0$ $\quad$ $1 \times 0 = 0$

$0 \times 1 = 0$ $\quad$ $1 \times 1 = 1$

(34) SCL b m — Selective Clear

Individual bits of A are set to 0's where there are corresponding 1's in the word at memory location M. If the corresponding bits in M are 0's, the associated bits of A remain unchanged.

(35) SST b m — Selective Set

Individual bits of A are set to 1's where there are corresponding 1's in the word at memory location M. If the corresponding bits in M are 0's, the associated bits of A remain unchanged.

(37) LSS b m — Logical Selective Set

Individual bits of A are set to 1's where there are corresponding 1's in the logical product of M and Q. If corresponding bits in the logical product of M and Q are zeros, the associated bits of A remain unchanged.

(57.0)
(57.1) CPA s t — Complement A

This instruction complements the contents of the A register. The address portion of this instruction, m, is not used. The designator, b, is ignored. The suboperation code, s, equals zero and specifies the A register for content complementing. The, t, bit specifies either one's or two's complement.

(57.2)
(57.3) CPQ s t — Complement Q

This instruction complements the contents of the Q register. The address portion of this instruction, m, is not used. The designator, b, is ignored. The suboperation code, s, equals one and specifies the Q register for content complementing. The, t, bit specifies one's or two's complement.

4.1.5.4 Transfer Instructions

All modes of address modification apply to the full-word transmission instructions except for XAQ, ENI, LDI, STI, ATI, and QTI.

(14) LDA b m — Load A

Replaces the contents of A with a 24-bit operand contained in memory location M. The contents M remain unchanged.

(17) LDQ b m — Load Q

Replaces the contents of Q with a 24-bit operand contained in the memory location M. The contents of M remain unchanged.

(56) LAC b m — Load A Complement

Replaces the contents of A with the two's complement of the 24-bit operand contained in memory location M.

(10) STA b m — Store A

Replaces the contents of the designated storage location, M, with the contents of A. The initial contents of A remain unchanged.

(13) STQ b m — Store Q

Replaces the contents of the designated storage location, M, with the contents of Q. The initial contents of Q remain unchanged.

(67) XAQ b m — Exchange A and Q

Places the initial contents of the A register into the Q register and at the same time places the initial contents of the Q register into the A register. The result of this instruction is that the contents of the A and Q registers are interchanged. The m and b portions of this instruction are not used.

(52) ENA b y — Enter A

The 15-bit operand, Y, is entered into the A register and its highest-order bit (sign bit) is extended in the remaining nine bits. The largest positive 15-bit operand that can be entered into A is 37777_8 $(2^{14}-1)$ and the 0 sign bit will be duplicated in each of the upper nine bits.

(53) ENQ b y — Enter Q

The 15-bit operand, Y, is entered into the Q register. The same conditions as for the ENA instruction are applicable.

(60) ENI b y — Enter Index

Replaces (B^b) with the operand y. If b = 0, 5, 6, or 7, this instruction becomes a pass or do-nothing instruction.

(15) LAL b m — Load A Left

Replaces the contents of A with a 24-bit operand. The left half of the 24-bit operand is obtained from the left half of memory location M. The right half of the 24-bit operand is zero bits. The contents of M remain unchanged.

(16) LAR b m — Load A Right

Replaces the contents of A with a 24-bit operand. The left half of the 24-bit operand is obtained from the right half of memory location M. The right half of the 24-bit operand is zero bits. The contents of M remain unchanged.

(11) SAL b m — Store A Left

Replaces the left half of storage location M with left half of A. The initial contents of A and the right half of storage location M remain unchanged.

(12) SAR b m — Store A Right

Replaces the right half of storage location M with the left half of A. The initial contents of A and the left half of storage location M remain unchanged.

(55) STZ b m — Store Zero

Replaces the contents of storage location M with zero. The initial contents of A and Q remain unchanged.

(64) LDI b m — Load Index

Replaces the contents of the designated index register with the address portion of memory location m. If b = 0, 5, 6, or 7, this instruction becomes a pass or do-nothing instruction. Initial contents of m remain unchanged.

(54) STI b m — Store Index

Replaces the address portion of storage location m with the contents of the designated index register. The remaining bits of the word in storage are set to 0. If b = 0, 5, 6 or 7, the word is set to zero.

(65) ATI b — A to Index

This instruction transmits the lower-order 15 bits of the A register to the index register specified by the index designator. The address portion of the instruction, m, is not used in this instruction. The contents of the A register are unchanged by this instruction. If b = 0, 5, 6, or 7, this instruction becomes a pass or do-nothing instruction.

(66) QTI b — Q to Index

This instruction transmits the lower-order 15 bits of the Q register to the index register specified by the index designator. The address portion of the instruction, m, is not used. The contents of the Q register are unchanged as a result of this instruction. If b = 0, 5, 6, or 7, this instruction becomes a pass or do-nothing instruction.

4.1.5.5 Storage Search

If b = 0, in the following instructions, only the word at storage location m is searched. If b = 5, 6, or 7, or $(B^b) = 0$, no search is made. In all of these instructions a test is made for interrupt conditions after each word is searched.

(24) EQS b m — Equality Search

Searches a list of operands to find one that is equal to (A). The number of items to be searched is specified by (B^b). These items are in sequential addresses beginning at location m + 1. The search begins with the last address, m + (B^b). (B^b) is reduced one count for each word that is searched until an operand is found that equals (A) or until (B^b) equals zero. If the search is terminated by finding an operand that equals (A), the next instruction is skipped. The address of the operand satisfying this condition is given by the sum of m + 1 and the final contents of B^b. If no operand is found that equals (A), the next instruction is taken.

(25) THS b m — Threshold Search

Searches a list of operands to find one that is greater than (A). The number of items to be searched is specified by (B^b). These items are located in sequential addresses beginning at location m + 1. The search begins with the last address, m + (B^b). The content

of the index register is reduced by one for each operand examined. The search continues until an operand is reached that is greater than (A) or until (B^b) is reduced to zero. If the search is terminated by finding an operand greater than the value in A, the next instruction is skipped. The address of the operand satisfying the condition is given by the sum of m + 1 and the final contents of B^b. If no operand in the list is greater than the value of A, the next instruction is taken.

(26) MEQ b m — Masked Equality

Searches a list of operands to find one operand in which the logical product of (Q) and (M) is equal to (A). This instruction, except for the mask in Q, operates in the same manner as an equality search.

(27) MTH b m — Masked Threshold

Searches a list of operands to find one operand in which the logical product of (Q) and (M) is greater than (A). Except for the mask in Q, this instruction operates in the same manner as the threshold search.

Figure 4-2 illustrates the four search instructions just described.

4.1.5.6 Jumps and Stops

Address modification is used in the UJP and JPI instructions only. A jump instruction causes a current program sequence to terminate and initiates a new sequence at a different location in storage. The program address register, P, provides the continuity between program steps and always contains the storage location of the current program step.

When a jump instruction occurs, P is cleared and a new address is entered into P. In most jump instructions, the base execution address, m, is entered to specify the new address for beginning the new program sequence. The word at location m is read from storage, placed in U, and the first instruction of the new sequence is executed.

Some of the jump instructions are conditional upon a register containing a specific value or upon the condition of the SE stop switch. If the criterion is satisfied, the jump is made to location m. If it is not satisfied, the program proceeds in its regular sequence to the next instruction.

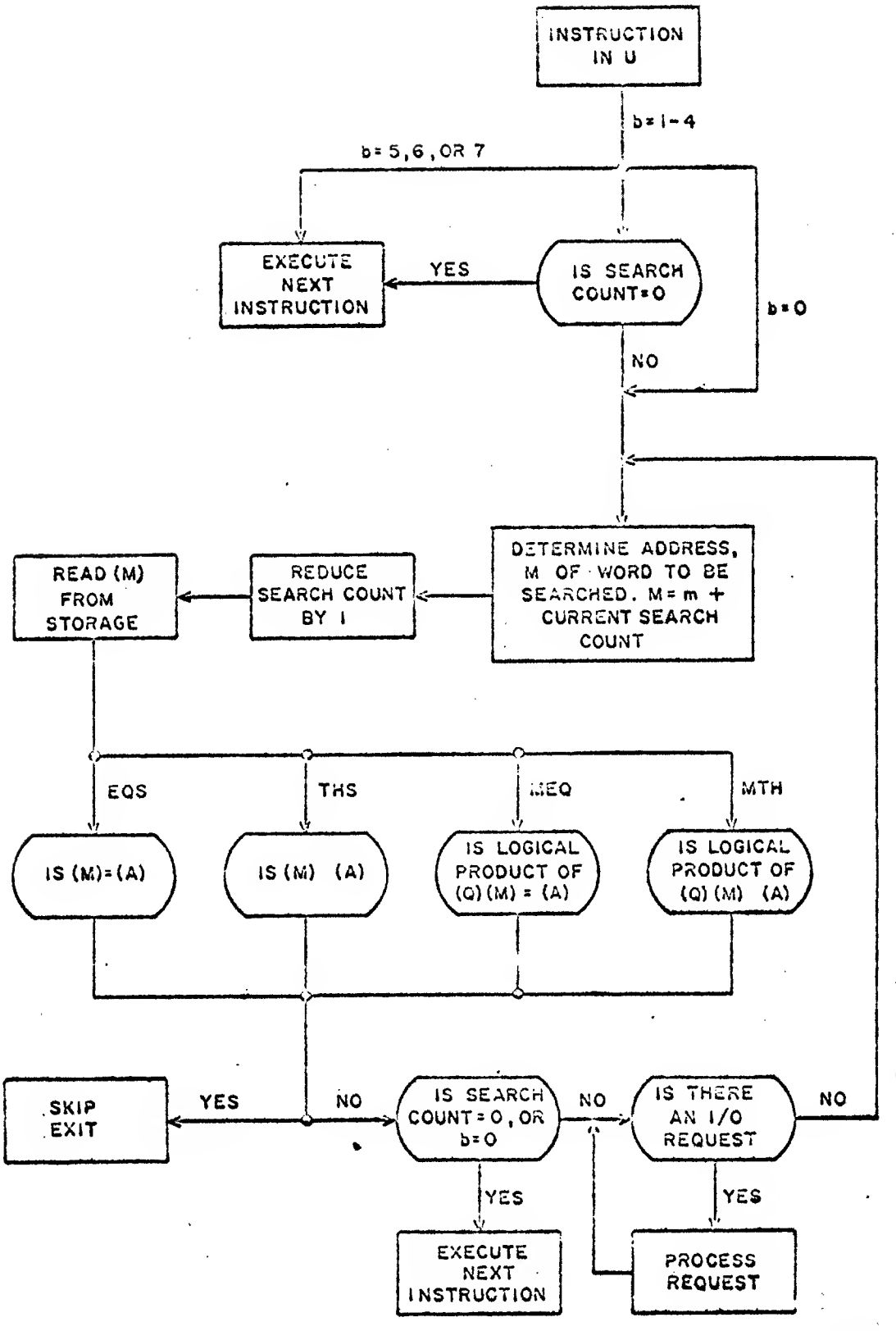


Figure 4-2. Flow Diagram of Search Instructions

(72) AJP j m — A Jump

Jumps to m if the condition of the A register specified by the jump designator j exists. If not, the next instruction is executed.

j = 0	Jump if (A) = 0	j = 6	Jump if $(A_R) \geq 0$
j = 1	Jump if (A) $\neq$ 0	j = 7	Jump if $(A_R) < 0$
j = 2	Jump if (A) $\geq$ 0		
j = 3	Jump if (A) < 0		
j = 4	Jump if $(A_R) = 0$		
j = 5	Jump if $(A_R) \neq 0$		

(70) SLJ j m — Selective Jump

Stops at the present step in the sequence if the condition specified by j exists. Then jumps to m if the condition specified by j exists.

The next instruction is executed if the jump condition does not exist.

j = 0	Stop if SE connected; jump unconditionally.
j = 1	Stop if SE connected; jump upon restart if jump switch set.
j = 2	Stop if stop switch is set; jump unconditionally.
j = 3	Stop if stop switch is set; jump if jump switch set.

(73) AJS j m — A Jump Stop

Stops at the present step in the sequence if the stop switch is set. Jump to m if the condition of the A register specified by the jump designator j exists.

j = 0	Jump if (A) = 0	j = 4	Jump if $(A_R) = 0$
j = 1	Jump if (A) $\neq$ 0	j = 5	Jump if $(A_R) \neq 0$
j = 2	Jump if (A) $\geq$ 0	j = 6	Jump if $(A_R) \geq 0$
j = 3	Jump if (A) < 0	j = 7	Jump if $(A_R) < 0$

(62) IJP b m — Index Jump

Examines (B^b). If this quantity is not zero, the quantity is reduced one count and a jump is executed to program step m. IJP terminates at zero. If (B^b) is zero, the present program sequence is continued. If b = 0, 5, 6, or 7, this instruction becomes a pass or do-nothing instruction.

(71) UJP b m — Unconditional Jump

Jumps unconditionally to address M. $M = m + (B^b)$. If b = 0, 5, 6, or 7, M = m for the jump.

(75) JPI b m — Jump Indirect

Jumps unconditionally to address specified by the contents of M. If b = 0, 5, 6, or 7, M = m for the jump. If b = 7, the interrupt inhibit bit in the interrupt mask register is also cleared. The JPI instruction differs from the UJP in that the contents of the address specified by M is read to obtain the jump address.

(76) SRJ j m — Subroutine Jump

Replaces the contents of storage location designated by 2000_8 + (j), $0 \leq j \leq 7$, with the address of the next instruction, $(P) + 1$, then jumps unconditionally to m. A subroutine jump, Figure 4-3, begins a new program sequence at the address specified by m, for example, 10100. At the same time, the lower 15 bits (execution address) of storage location 002001 are replaced with the address of the next program step in the main program 10011. The last instruction in a new program sequence is a Jump Indirect instruction which enters m stored in 02001 into P to continue the main program sequence.

(74) JSI b m — Jump Unconditional and Set Index

Jumps unconditionally to m. At the same time the (P) + 1 is placed in index register B^b. If b = 0, 5, 6, or 7, only a jump to m is executed.

(63) ISK b y — Index Skip

Compares (B^b) with y. If the two quantities are equal, B^b is cleared and the next instruction is skipped. If the quantities are unequal, (B^b) is increased one count and the next instruction is taken. If b = 0, 5,6, or 7, this instruction becomes a pass or do-nothing instruction.

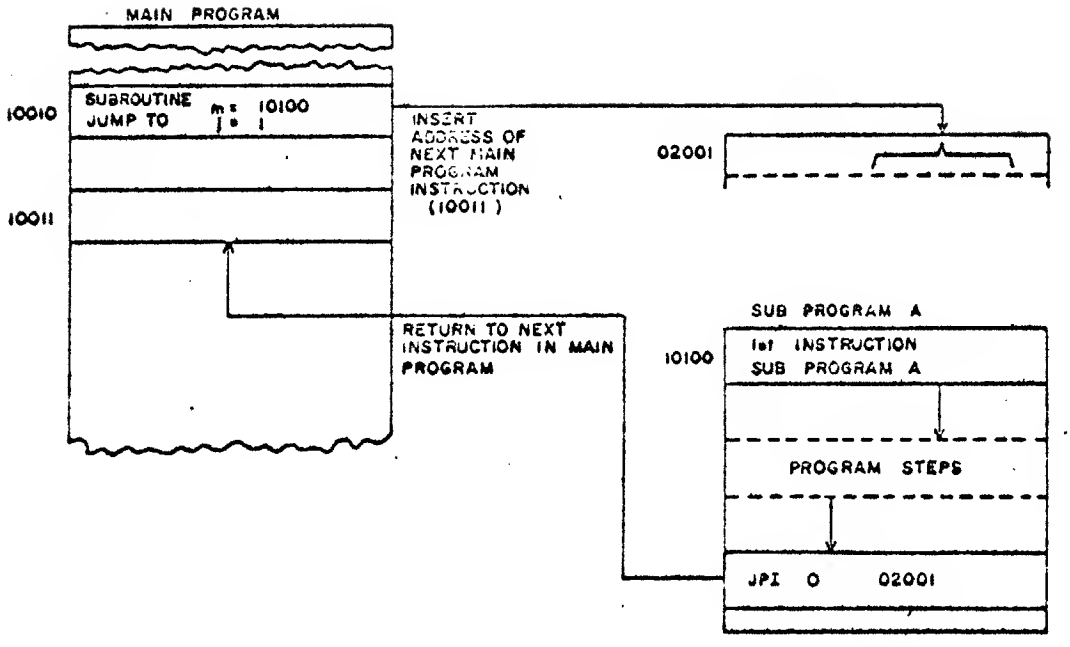


Figure 4-3. Subroutine Jump Instruction

4.2 MISCELLANEOUS INSTRUCTIONS

(77) INT J — Interrupt

The INT instructions is forced into the U register when an interrupt is recognized. The execution of this instruction is similar to the SRT except that the location of the next instruction, (P)+1, is stored at location 2010+j.

Example: The external interrupt forces the instruction INT 6 77776 into the U register where it is executed as the next instruction. (P)+1 is stored at location 02016_8 since j=6. Then a jump is taken to 37776, the lower 14 bits of 77776.

(40) NO P — No Operation

This instruction is a "do nothing" operation. No registers are altered and the instruction is effectively a 2.5 microsecond delay in the program execution.

EXF — External Function

The External Function instructions (EXF) perform four types of normal operations (see Table 4-2).

1) Activate - activates a channel input/output operation.

   The EXF activate instruction (j=1 or 5, c=02, d=data address), loads the program controlled data address into the data address register and activates the input or output channel by sending the strobe input (or output) address signal to the external equipment.

2) Transfer - performs a data transfer.

   The EXF Transfer instruction (j=0,1,4,5,c=01, d=not used) transfers data from the external lines to the A register or from the A register to the output register. The transfer instruction also loads the interrupt mask register with the contents of A(j=0, c=04, d=not used) and allows reading the external interrupt register into A(j=1, c=4, d=not used).

3) Sense - senses external condition.

The EXF Sense instruction provides a means for testing the system channel input ready line (j=2, c=not used, d=not used) and also for testing support equipment discretes (j=6, c=not used, d=support equipment discretes).

4) Function - selects a function.

Certain functions can also be executed by means of the EXF instructions. These functions are:

a) Set Console Function(j=4, c=10, d=support equipment functions). This function selects peripheral equipment on the support equipment channel.

b) Set Interrupt Lockout(j=0, c=20, d=not used)

c) Clear Interrupt Lockout(j=0, c=40, d=not used)

4.2.1 I/O SEQUENCE

The input and output operations must be performed in the following sequences:

4.2.1.1 Output

Step	Instruction	Operation
1	LDA	Load A - Output data transferred to A
2	EXF,j=0,c=01 d=not used	A transferred to the output register and the output lines
3	EXF,j=0,c=02 d=data address	Data address and the output address strobe are transmitted to the external equipment

The external functions in the output routine can be determined using Table 4-3. An explantion of the output operation follows. Numbers in parenthesis refer to item numbers in Table 4-3. Step 1 requires the program to load the output data into the A register. In Step 2 (EXF,j=0, c=01, d=not used) the program executes an EXF data transfer instruction which transmits the

contents of A to the output register where the data is placed on the output data lines ((A register)→X register (1) and (X register)→PCC output register (10))

In step 3 (EXF, j=0, c=02, d=data address), the program executes an EXF activate output channel which transmits the data address to the external equipment along with an output address strobe signal. The external equipment detects the output address strobe signal and reads the output address and the output data. ((A register)→X register (1) and PCC output strobe→I/O unit (13)). The output function is then completed.

4.2.1.2 Input

Step	Instruction	Operation
1	EXF,j=1, c=02, d=data address	Data address and input address strobe are transmitted to the external equipment
2	EXF,j=2, c=not used d=not used	Sense input data ready. If input data ready detected skip step 3.
3	UJP	Jump to step 2.
4	EXF,j=1, c=01 d=not used	Input data transfer to A.

The preceding input operation uses combinations of function codes from Table 4-3. In the following explanation numbers in parentheses denote item numbers in Table 4-3.

Step 1 (EXF,j=1, c=02, d=data address) requires the computer program to transmit to the external equipment a data address which specifies the data word to be used as input. An input address strobe signal is simultaneously transmitted to the external equipment. This operation is performed by the execution of an EXF instruction, activate input channel. The external equipment detects the input address strobe, reads the input address, places the specified data on the input data lines, and transmits an input ready signal to the computer. ((X register)→A register (2) and PCC input address strobe→I/O unit (14)).

The second step tests the input ready line for the presence of the input ready signal by means of an EXF sense instruction. (Sense PCC input data ready (4)).

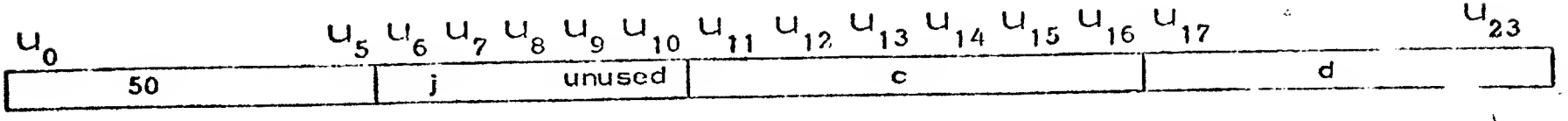
When the input ready signal is detected by the EXF sense instruction, a skip condition occurs. Step 3 causes a loop to await an input ready signal.

In step 4 (EXF, j=1, c=01, d=not used) the program executes an EXF data transfer instruction and the data on the input lines is loaded into the A register. (PCC input data→X register (5) and (X register)→A register (2)). Thus, the input sequence is completed.

4.2.2 EXF Format

Various functions in Table 4-3 have been combined to form the more useful functions in Table 4-2 for special purposes, other useful function codes can be derived from Table 4-3.

TABLE 4-2. EXTERNAL FUNCTION INSTRUCTION FORMAT



j Function		c Sub function		d (PID/POT)
0	Output to Program Control Channel (24 bits)	01	Data transfer (j=0,1,4,5)	Not used
1	Input from Program Control Channel (24 bits)	02	Activate channel (j=0,1)	Data Address
2	Sense Input Data Read (Program Control Channel)	04	(A)—> Mask Register (j=0) Interrupts —> A (j=1)	Not used
4	Output to Support Equipment Channel (7 bits)	10	Set Console Function (j=4)	Support equipment
5	Input from Support Equipment Channel (8 bits)	20	Set Interrupt Lockout (j=0)	Not used
6	Sense Support Equipment discretes (7 bits)	40	Clear Interrupt Lockout (j=0)	Not used
		Not used	Sense Input Ready line	Not used
		Not used	Sense Support Equipment discretes	Support Equipment discretes

*TABLE 4-3. EXF FUNCTIONS

Item	Function
1	If $\overline{U}_6$, A Register → X Register
2	If U_8, X Register → A Register
3	If U_6U_7, Sense Console discretes
4	If $\overline{U}_6U_7$, Sense PCC Input Data Ready
5	If $\overline{U}_6U_8U_{16}$, PCC Input Data → X Register
6	If U_8U_{14}, Interrupts → X, Interrupt Status Clear → I/O Unit
7	If $U_6U_8U_{16}$, Console Data → X Register
8	If $\overline{U}_6U_{15}$, U_{17-23} → PCC Data Address Register
9	If $\overline{U}_6\overline{U}_8U_{14}$, X Register → Interrupt Mask Register
10	If $\overline{U}_6\overline{U}_8U_{16}$, X Register → PCC Output Register
11	If $U_6\overline{U}_8$, Typewriter Data Ready → Console
12	If U_6U_{13}, Select Console Function → Console
13	If $\overline{U}_6\overline{U}_8U_{15}$, PCC Output Address Strobe → I/O Unit
14	If $\overline{U}_6U_8U_{15}$, PCC Input Address Strobe → I/O Unit
15	If U_{12}, Set Interrupt Inhibit FF
16	If U_{11}, Clear Interrupt Inhibit FF

* U_x = instruction format with x being a bit position

$0 \leq x \leq 23$, beginning with U_0 as the most left bit position

$\overline{U}_x$ = bit unset (0) and U_x = set (1). See diagram in Table 4-3.

APPENDIX A

REPERTOIRE OF INSTRUCTIONS

Table A-1 contains a description of the instructions. Table A-2 lists the instructions in alphabetical order of the mnemonic.

TABLE A-1.
DESCRIPTION AND EXECUTION TIME OF INSTRUCTIONS

Op Code	Mnemonic	Name	Description	Execution Time
00	ADD	Add	$(A)+(M) \rightarrow A$	2.5
01	ADL	Add Left	$(A_L)+(M_L) \rightarrow A_L$, A_R unchanged	2.5
02	ADR	Add Right	$(A_L)+(M_R) \rightarrow A_L$, A_R unchanged	
03	ADQ	Add to Q	$(Q)+(M) \rightarrow Q$	2.5
04	SUB	Subtract	$(A)-(M) \rightarrow A$	2.5
05	SBL	Subtract Left	$(A_L)-(M_L) \rightarrow A_L$, A_R unchanged	2.5
06	SBR	Subtract Right	$(A_L)-(M_R) \rightarrow A_L$, A_R unchanged	2.5
07	SBQ	Subtract from Q	$(Q)-(M) \rightarrow Q$	2.5
10	STA	Store A	$(A) \rightarrow M$	2.5
11	SAL	Store A Left	$(A_L) \rightarrow M_L$, (M_R) unchanged	2.5

TABLE A-1.
DESCRIPTION AND EXECUTION TIME OF INSTRUCTIONS (Cont.)

Op Code	Mnemonic	Name	Description	Execution Time
12	SAR	Store A Right	$(A_L) \rightarrow M_R$, (M_L) unchanged	2.5
13	STQ	Store Q	$(Q) \rightarrow M$	2.5
14	LDA	Load A	$(M) \rightarrow A$	2.5
15	LAL	Load A Left	$(M_L) \rightarrow A_L, 0 \rightarrow A_R$	2.5
16	LAR	Load A Right	$(M_R) \rightarrow A_L, 0 \rightarrow A_R$	2.5
17	LDQ	Load Q	$(M) \rightarrow Q$	2.5
20	MUF	Multiply Fractional	$(A) \times (M) \rightarrow AQ$	25.0 1/
21	MFL	Multiply Fractional Left	$(A) \times (M_L) \rightarrow A$ and Q_L	15.0 1/
22	MFR	Multiply Fractional Right	$(A) \times (M_R) \rightarrow A$ and Q_L	15.0 1/
23	DVF	Divide Fractional	$(AQ) \div (M) \rightarrow Q$ remainder $\rightarrow A$	27.5 1/
24	EQS	Equality Search	Search (B^b) words until $(M)=(A)$	2.5+2.5n 1/
25	THS	Threshold Search	Search (B^b) words until $(M) \geq A$	2.5+2.5n 1/

1/ This includes the access time for the next instruction. If the jump is not taken in the AJP, AJS, and SLJ instructions the execution time is 2.5 microseconds.

TABLE A-1.
DESCRIPTION AND EXECUTION TIME OF INSTRUCTIONS (Cont.)

Op Code	Mnemonic	Name	Description	Execution Time
26	MEQ	Masked Equality Search	Search (B^b) words until (Q) ∘ (M) = (A)	$2.5+2.5\frac{1}{n}$
27	MTH	Masked Threshold Search	Search (B^b) words until (Q) ∘ (M) > (A)	$2.5+2.5\frac{1}{n}$
30	RAD	Replace Add	(A) + (M) → A and M	5.0
31	RSB	Replace Subtract	(A) - (M) → A and M	5.0
32	RAØ	Replace Add One	(M) + 1 → A and M	5.0
33	RSØ	Replace Subtract One	(M) - 1 → A and M	5.0
34	SCL	Selective Clear	0 → A_N, if $(M_N)=1$	2.5
35	SST	Selective Set	1 → A_N, if $(M_N)=1$	2.5
36	LDL	Load Logical Product	(Q) ∘ (M) → A	2.5
37	LSS	Logical Selective Set	1 → A_N, if (M_N) ∘ $(Q_N)=1$	2.5
40	NØP	No Operation		2.5
41.0	QRS	Q Right Shift, Sign Extended	Shift (Q) right K places	$2.5+\left[\frac{K+1}{3}\right]\times 2.5$
41.1	QRE	Q Right Shift, End Off	Shift (Q) right K places	$2.5+\left[\frac{K+1}{3}\right]\times 2.5$

TABLE A-1.
DESCRIPTION AND EXECUTION TIME OF INSTRUCTIONS (Cont.)

Op Code	Mnemonic	Name	Description	Execution Time
42.0	ARS	A Right Shift, Sign Extended	Shift (A) right K places	$2.5+\left[\frac{K+1}{3}\right]\times 2.5$
42.1	ARE	A Right Shift, End Off	Shift (A) right K places	$2.5+\left[\frac{K+1}{3}\right]\times 2.5$
43.0	LRS	Long Right Shift, Sign Extended	Shift (AQ) right K places	$2.5+\left[\frac{K+1}{3}\right]\times 2.5$
43.1	LRE	Long Right Shift, End Off	Shift (AQ) right K places	$2.5+\left[\frac{K+1}{3}\right]\times 2.5$
44	SCA	Scale A	Shift (A) left until $(A_0)\neq(A_1)$ or k-no. of shifts=0; k-no. of shifts→B	$2.5\pm\left[\frac{\text{no. of shifts}+1}{3}\right]\times 2.5$
45.0	QLC	Q Left Shift, Circular	Shift (Q) left K places	$2.5+\left[\frac{K+1}{3}\right]\times 2.5$
45.1	QLE	Q Left Shift, End Off	Shift (Q) left K places	$2.5\pm\left[\frac{K+1}{3}\right]\times 2.5$
46.0	ALC	A Left Shift, Circular	Shift (A) left K places	$2.5+\left[\frac{K+1}{3}\right]\times 2.5$
46.1	ALE	A Left Shift, End Off	Shift (A) left K places	$2.5+\left[\frac{K+1}{3}\right]\times 2.5$

TABLE A-1.
DESCRIPTION AND EXECUTION TIME OF INSTRUCTIONS (Cont.)

Op Code	Mnemonic	Name	Description	Execution Time
47.0	LLC	Long Left Shift, Circular	Shift (AQ) left K places	$2.5+\left[\frac{K+1}{3}\right]\times 2.5$
47.1	LLE	Long Left Shift, End Off	Shift (AQ) left K places	$2.5+\left[\frac{K+1}{3}\right]\times 2.5$
50	EXF	External Function		2.5 - 5.0
51	INA	Increase A	(A)+Y→ A	2.5
52	ENA	Enter A	Y→A	2.5
53	ENQ	Enter Q	Y→ Q	2.5
54	STI	Store Index	(B^b)→ m; if b=0, 5, 6, or 7, then 0→ m	2.5
55	STZ	Store Zero	0→ M	2.5
56	LAC	Load A Twos, Complement	$\overline{(M)}+1$→ A	2.5
57.0	CPA,1	Complement A, Ones Complement	$\overline{(A)}$ → A	2.5
57.1	CPA,2	Complement A, Twos Complement	$\overline{(A)}+1$→ A	2.5
57.2	CPQ,1	Complement Q, Ones Complement	$\overline{(Q)}$ → Q	2.5

TABLE A-1.

DESCRIPTION AND EXECUTION TIME OF INSTRUCTIONS (Cont.)

Op Code	Mnemonic	Name	Description	Execution Time
57.3	CPQ,2	Complement Q, Twos Complement	$(\overline{Q})+1 \rightarrow Q$	2.5
60	ENI	Enter Index	$y \rightarrow B^b$	2.5
61	INI	Increase Index	$(B^b)+y \rightarrow B^b$	2.5
62	IJP	Index Jump	If $(B^b) \neq 0$, $(B^b)-1 \rightarrow B^b$ and jump to m	5.0 1/
			If $(B^b)=0$, take N.I.	2.5
63	ISK	Index Skip	If $(B^b) \neq y$, $(B^b)+1 \rightarrow B^b$ take N.I.; If $(B^b)=y$, $0 \rightarrow B^b$ & skip N.I.	5.0 1/ 2.5
64	LDI	Load Index	$(M_{9-23}) \rightarrow B^b$	2.5
65	ATI	A to Index	$(A_{9-23}) \rightarrow B^b$	2.5
66	QTI	Q to Index	$(Q_{9-23}) \rightarrow B^b$	2.5
67	XAQ	Exchange A and Q	$(A)_i \rightarrow Q$ $(Q)_i \rightarrow A$	2.5
70	SLJ	Selective Jump	Jump to m if condition j	5.0 1/
71	UJP	Unconditional Jump	Jump to M unconditionally	5.0 1/

1/ This includes the access time for the next instruction. If the jump is not taken in the AJP, AJS, and SLJ instructions the execution time is 2.5 microseconds.

TABLE A-1.
DESCRIPTION AND EXECUTION TIME OF INSTRUCTIONS (Cont.)

Op Code	Mnemonic	Name	Description	Execution Time
72	AJP	A Jump	Jump to m if condition j	5.0 1/
73	AJS	A Jump Stop	Stop if stop switch set. Jump to m if condition j.	5.0 1/
74	JSI	Jump Unconditionally Set Index	(P)+1 → B^b and Jump to m	5.0 1/
75	JPI	Jump Indirect	Jump to (m); if b=7 clear interrupt inhibit bit	5.0 1/
76	SRJ	Subroutine Jump	(P)+1→(02000_8+j); Jump to m	5.0 1/
77	INT	Interrupt	(P)+1→ (02000_8+j); Jump to m	5.0 1/

1/ This includes the access time for the next instruction. If the jump is not taken in the AJP, AJS, and SLJ instructions the execution time is 2.5 microseconds.

TABLE A-2.
LIST OF INSTRUCTIONS IN ALPHABETICAL ORDER OF MNEMONIC

OP Code	Mnemonic	Name	OP Code	Mnemonic	Name
00	ADD	Add	52	ENA	Enter A
01	ADL	Add Left	60	ENI	Enter Index
03	ADQ	Add to Q	53	ENQ	Enter Q
02	ADR	Add Right	24	EQS	Equality Search
72	AJP	A Jump	50	EXF	External Function
73	AJS	A Jump Stop	62	IJP	Index Jump
46.0	ALC	A Left Shift Circular	51	INA	Increase A
46.1	ALE	A Left Shift, End Off	61	INI	Increase Index
42.1	ARE	A Right Shift, End Off	77	INT	Interrupt
42.0	ARS	A Right Shift, Sign Extended	63	ISK	Index Skip
65	ATI	A to Index	75	JPI	Jump Indirect
57.0	CPA,1	Complement A, Ones Complement	74	JSI	Jump and Set Index
57.1	CPA,2	Complement A, Twos Complement	56	LAC	Load A twos Complement
57.2	CPQ,1	Complement Q, Ones Complemer	15	LAL	Load A Left
57.3	CPQ,2	Complement Q, Twos Complement	16	LAR	Load A Right
23	DVF	Divide Fractional	14	LDA	Load A

TABLE A-2.
LIST OF INSTRUCTIONS IN ALPHABETICAL ORDER OF MNEMONIC (Cont.)

OP Code	Mnemonic	Name	OP Code	Mnemonic	Name
64	LDI	Load Index	30	RAD	Replace Add
36	LDL	Load Logical Product	32	RAØ	Replace Add One
17	LDQ	Load Q	31	RSB	Replace Subtract
47.0	LLC	Long Left Shift, Circular	33	RSØ	Replace Subtract One
47.1	LLE	Long Left Shift, End Off	11	SAL	Store A Left
43.1	LRE	Long Right Shift, End Off	12	SAR	Store A Right
43.0	LRS	Long Right Shift, Sign Extended	05	SBL	Subtract Left
37	LSS	Logical Selective Set	07	SBQ	Subtract from Q
26	MEQ	Masked Equality Search	06	SBR	Subtract Right
21	MFL	Multiply Fractional Left	44	SCA	Scale A
22	MFR	Multiply Fractional Right	34	SCL	Selective Clear
27	MTH	Masked Threshold Search	70	SLJ	Selective Jump
20	MUF	Multiply Fractional	76	SRJ	Subroutine Jump
40	NØP	No Operation	35	SST	Selective Set
45.0	QLC	Q Left Shift, Circular	10	STA	Store A
45.1	QLE	Q Left Shift, End Off	54	STI	Store Index
41.1	QRE	Q Right Shift, End Off	13	STQ	Store Q
41.0	QRS	Q Right Shift, Sign Extended	55	STZ	Store Z
66	QTI	Q to Index	04	SUB	Subtract

TABLE A-2.
LIST OF INSTRUCTIONS IN ALPHABETICAL ORDER OF MNEMONIC (Cont.)

OP Code	Mnemonic	Name	
25	THS	Threshold Search	
71	UJP	Unconditional Jump	
67	XAQ	Exchange A and Q	


CONTROL DATA
CORPORATION